**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»**

**Факультет информатики**

**Кафедра технической кибернетики**

**Самостоятельная работа №2**

**по теме: *“Ошибки в программе”*.**

Выполнил: Борисов Д.С.

Проверил: Куприянов А.В.

Группа: 6110

САМАРА 2018

**СОДЕРЖАНИЕ**

1. Введение.
2. Основная часть.
3. Заключение
4. Литература.

**Введение**

Исправление ошибок – это неотъемлемая часть процесса разработки программы, на которую иногда требуется гораздо больше времени, чем само написание кода, ведь редко когда удаётся написать идеальную программу с первого раза и добиться того, чего хотел. В этом нет ничего постыдного, ведь даже профессионалы ошибаются, а учиться исправлять свои или же чужие ошибки – тоже опыт.

Источники ошибок бывают нескольких видов:

1. Плохая спецификация (Что должен делать тот или иной блок кода?)
2. Незаконченные программы («Напишу как-нибудь потом»)
3. «Неожиданные» значения входных параметров (Деление на ноль и другие)
4. Ошибки реализации

Все ошибки можно разделить на несколько типов:

1. Ошибки времени компиляции

* Синтаксические ошибки (пропущенные точки с запятой, фигурные скобки и другие; неправильные имена операторов и не только)
* Ошибки типизации (несоответствие типа и значения)

1. Ошибки времени компоновки
2. Ошибки времени выполнения

* Обнаруживаемые системой (сбой программы)
* Обнаруживаемые библиотеками (исключения)
* Обнаруживаемые пользовательским кодом

1. Логические ошибки

* Обнаруживаемые человеком (программа выдаёт не тот результат, который ожидал пользователь)

На мой взгляд, самое трудное в программировании – это донести до компилятора то, чего ты от него хочешь, ведь он не может додумать ход ваших мыслей. Он будет выполнять исключительно то, что ему пропишут, а не то что вы имеете в виду. Как ни странно, это правильно, ведь если бы вы чётко выразили свою мысль в виде кода, а компилятор выполнил совсем другую команду, то написание программы стало бы мучительным процессом. Непредсказуемость действий машины стало бы большой проблемой в разработке. «Недопонимание» человека и компилятора вызывает **логическую ошибку**. Программа успешно проходит все стадии проверок, но выдаёт неверный результат. Мне кажется, обнаружение и исправление таких ошибок – наиболее сложная задача, поэтому данную проблему стоит рассмотреть внимательнее.

**Основная часть**

Логическая ошибка может произойти по нескольким причинам:

* **Ошибочное представление разработчика о действиях той или иной алгоритмической инструкции.** Например, он может подумать что в тернарном операторе *a = (b < c) ? 1 : 0* переменная *a* присвоит значение 0, если *b* будет меньше *c*;
* **Отсутствие фрагмента кода.** Разработчик не написал необходимые строки кода, однако без них программа смогла компилироваться;
* **Потеря или преобразование данных при некорректном приведении к типу.** Например, занесение в переменную типа int значение с плавающей точкой:

int a = 5/2; //Будет занесено 2, а не 2.5

* **Обыкновенная опечатка, не мешающая выполнению программы.** Данная проблема встречается очень часто, когда пользователь случайно вводит не те значения, которые хотел. Например, случайное нажатие на клавишу:

return 100 \* 45; //Разработчик случайно нажал на кнопку 5

* **Неправильное использование синтаксиса языка разработчиком, не мешающее выполнению программы.** Например, разработчик хочет в приведённом ниже коде 3 раза вывести на экран “Age: 10”, где значение 10 будет храниться в переменной *age*, а в результате выведется 3 раза “Age: ”, а только потом число 10.

for (int i = 0; i < 3; ++i)

cout << “Age: “;

cout << age;

Разумеется это не все причины, но их достаточно для того, чтобы разобраться в природе возникновения логических ошибок. Они появляются из-за того, что разработчик не всегда может корректно перевести ход своих мыслей на понятный для машины язык, в результате чего программа может вернуть неверный результат.

Поиск таких ошибок – это сложнейшая задача, на которую требуется гораздо больше времени, чем на обнаружение других проблем, ведь компилятор не сможет сам направить вас на источник ошибки и тогда её придётся искать самому.

**Заключение**

Я попытался рассказать о самой трудно распознаваемой и сложной ошибке при создании кода, чаще всего возникающей из-за невнимательности или из-за неопытности разработчика и обусловлена тем, что компилятор не может предсказать ход ваших мыслей, поэтому он будет выполнять только прописанный код и ничто иное.

Обнаружение и исправление логической ошибки можно назвать настоящим испытанием для программиста, после преодоления которого он учится думать и размышлять также как машина, что очень помогает при создании кода.

**Литература**

* Bjarne Stroustrup “Programming: Principles and Practice Using C++”, (Бьярне Страуструп, «Программирование: принципы и практика использования C++»)
* https://ru.wikipedia.org/wiki/Логическая\_ошибка\_(программирование)